Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tilen Tomakić

Angular in SPA – varna izmenjava podatkov med uporabniki

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentorica: doc. dr. Mira Trebar

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Varnost podatkov je v internetnem okolju zelo pomembna zahteva, ki jo je potrebno upoštevati pri razvoju spletnih aplikacij. Kandidat naj razišče možnosti za zaščito podatkov z uporabo kriptografije. Rešitev naj zasnuje za različne načine šifriranja podatkov v spletni aplikaciji na strani odjemalca. Predlagani pristop za varno izmenjavo podatkov med uporabniki naj implementira z ogrodjem Angular in tehnologijo izvajanja aplikacij na strani odjemalca SPA (Single-page application) ter oceni njegove prednosti in pomanjkljivosti.

Hvala mentorici, doc. dr. Miri Trebar, za pomoč in konstruktivno kritiko.

Zahvaljujem se tudi podjetju Kalmia, d.o.o., za podporo pri razvoju spletne aplikacije KalPass.

Še posebej se zahvaljujem staršem, za spodbudo in motivacijo tekom celotnega študija.

Kazalo

Seznam kratic

kratica	angleško	slovensko
AES	Advanced Encryption Algorithm: Rijndael	Napredni šifrirni algoritem
API	Application programming interface	Vmesnik za programiranje aplikacij
CRUD	Create, read, update and delete	Izdelaj, beri, posodobi in izbriši
DES	Data Encryption Standard	Standard za šifriranje podatkov
DMZ	Demilitarized Zone	Demilitarizirano območje
ECC	Elliptic curve cryptography	Šifriranje z eliptično krivuljo
ECDSA	Elliptic Curve Digital Signature Algorithm	Digitalno podpisovanje z eliptično krivuljo
IPsec	Internet Protocol Security	Internetni protokol varnosti
IoT	Internet of things	Internet stvari
JSON	JavaScript Object Notation	Format JavaScript za zapis objektov
JWT	JSON Web Token	Spletni žeton JSON
REST	Representational state transfer	Reprezentacijski prenos stanja
NIST	US National Institute of Standards and Technology	Ameriški Nacionalni inštitut za standarde in tehnologijo
SPA	Single-page application	Enostranska aplikacija
SSH	Secure Shell	Protokol varovane lupine
SSL	Secure Sockets Layer	Protokol za varno komunikacijo
XSS	Cross-site Scripting	Križno izvajanje skript

Povzetek

Naslov: Angular in SPA – varna izmenjava podatkov med uporabniki

Avtor: Tilen Tomakić

Uporabniki spletnih aplikacij nimajo druge izbire kot zaupati zalednim storitvam za oblačno hrambo podatkov, da bodo varovale njihove podatke. Pred vdori jih je v praksi praktično nemogoče zaščititi. V diplomskem delu je predstavljena možnost šifriranja (angl. *encryption*) podatkov znotraj spletne aplikacije. Taka rešitev prenese zaščito podatkov iz zaledja (angl. *backend*) na stran odjemalca. S tem sta zagotovljeni varnost in integriteta podatkov, tudi če ima napadalec dostop do zaledne storitve. Podatkov ne more dešifrirati (angl. *decrypt*) ali spreminjati, ker zaledne storitve ne potrebujejo ključa. Opisana rešitev poleg zaščite omogoča tudi varno izmenjavo podatkov. V ta namen je predstavljen postopek hrambe in izmenjave ključev med uporabniki. Rešitev je uporabljena v spletni aplikaciji KalPass. Zasnovana je na osnovi tehnologije SPA (angl. *Single-page application*) ob podpori ogrodja Angular. Predstavljen je proces zaščite in izmenjave podatkov, izpostavljene pa so tudi slabosti tovrstne zaščite v SPA.

Ključne besede: varna izmenjava podatkov, Angular, SPA, zaledna storitev, kriptografija, skrbnik gesel

Abstract

Title: Angular and SPA – secure data exchange between users

Author: Tilen Tomakić

Web application users have no other option than to trust backend services of cloud data storage that they will protect their data. In practice, it is actually impossible to protect them from web attacks. This thesis presents the possibility of encrypting data within a web application. The proposed solution shifts responsibility of protecting data from the backend to the client side. This ensures both, the security and the integrity of the data, even if the attacker has access to the backend service. Attackers cannot decrypt or modify data because the key is not available to the backend system. Beside the protection, this solution also offers secure data exchange among users specified as a process of storage and exchange of keys. In the end, the implementation of KalPass web application presents data protection and exchange by using SPA technology and Angular framework.

Keywords: secure data sharing, Angular, SPA, backend service, cryptografy, password manager

Poglavje 1

Uvod

1.1 Ozadje

Podjetja vedno pogosteje posegajo po različnih storitvah v oblaku. Vendar pa uporaba takih storitev poleg vseh prednosti prinaša tudi varnostne pomisleke. Oblačne storitve za delovanje običajno potrebujejo zaledno storitev za hrambo podatkov. Posledično podatki fizično niso več znotraj podjetja, ampak pri ponudniku oblačne storitve. Velikokrat niti ne vemo točno, v katerih državah ponudniki hranijo podatke. S tem popolnoma izgubimo nadzor nad shranjenimi podatki in ne vemo, kaj vse se lahko z njimi dogaja. Ponudnik oblačne storitve lahko spremeni pogoje uporabe (hrambe podatkov). Podatki so lahko hranjeni v čistopisu, če pa so šifrirani, je to pogosto izvedeno v zaledju oblačne storitve. Zato obstaja nevarnost, da ob vdoru v zaledje ponudnika oblačne storitve pride do odtujitve in uporabe podatkov.

Napadalci običajno zlorabijo hrošče v programski opremi ter tako pridobijo dostop do zaledja in posledično občutljivih podatkov. Včasih jim uspe pridobiti polni dostop do sistema (administratorske pravice), s tem pa tudi kakršnakoli rešitev šifriranja v sistemu ni več učinkovita. Prav tako ne smemo zanemariti zaposlenih pri ponudniku oblačne storitve. Administratorji lahko zaobidejo vse varnostne mehanizme in pridobijo polni

dostop do zaupnih podatkov uporabnikov, ki jih nikoli ne bi smeli videti. Razlogi za tako dejanje so lahko na primer radovednost, nezadovoljstvo z delovnim mestom, odpoved.

Alternativa storitvam v oblaku je postavitev storitve (zaledje skupaj s spletno aplikacijo) v lokalni infrastrukturi podjetja. Čeprav s tem pridobimo večji nadzor nad podatki, druge grožnje ostajajo. Še vedno obstajajo zunanje (spletni vdori) in notranje grožnje (zaposleni v podjetju).

Za kakršnokoli rešitev storitve gre, glavni problem ostaja enak. Če je nekomu omogočen neomejen dostop do shrambe na tak ali drugačen način, lahko pridobi zaupne podatke podjetja. Uporabniki (podjetja) spletnih aplikacij nimajo druge izbire kot zaupati zalednim storitvam (svojim ali oblačnim), da bodo varovale njihove podatke pred grožnjami. To pomeni, da zaledne storitve, zadolžene za hrambo podatkov spletne aplikacije, v celoti nosijo odgovornost zaščite.

1.2 Namen in prispevek diplomskega dela

Namen diplomskega dela je preučiti, kako prenesti vlogo varovanja podatkov iz zaledne storitve v spletno aplikacijo. Pojem zaledna storitev je v diplomskem delu opredeljen kot kakršnakoli storitev, zaledna ali oblačna, ki jo spletna aplikacija potrebuje za hrambo in zagotavljanje podatkov. Zaščita podatkov se izvaja v spletni aplikaciji na strani odjemalca. To je doseženo s šifriranjem in dešifriranjem podatkov. Na ta način zaledna storitev vidi le šifrirane podatke, ključa ne pozna, saj ga ne potrebuje. S prenosom vloge zaščite je rešen glavni problem potrebe po zaupanju v zaledno storitev.

Predvideni prispevek diplomskega dela vključuje naslednje:

- Opis uporabe kriptografije (angl. *cryptography*) za zaščito podatkov. Ker je ob šifriranju podatkov otežena izmenjava teh med uporabniki,

je poleg zaščite predstavljen tudi postopek varne izmenjave podatkov med uporabniki.

- Predstavitev vseh potrebnih korakov za varno implementacijo zaščite in izmenjave podatkov. Proces je predstavljen na podlagi razvoja spletne aplikacije KalPass.
- Izpostavitev slabosti in omejitev predlagane rešitve za spletne aplikacije SPA (angl. *Single-page application*) ter ugotovitev, v kolikšni meri je bil rešen glavni problem.

1.3 Struktura diplomskega dela

V drugem poglavju je na začetku kratek uvod v kriptografijo, definirano je ciljno okolje, za katero je rešitev zasnovana, nato sta opisani zaščita in izmenjava podatkov. V tretjem poglavju so predstavljene implementacija in pomanjkljivosti predlagane rešitve v SPA.

Poglavje 2

Varna izmenjava podatkov

2.1 Kriptografija

Kriptografija je znanost, ki se ukvarja z zapisom sporočila (čistopis) v taki obliki, da je njegov pomen skrit [1]. Sporočilo, katerega pomen vsebine ni znan, imenujemo tajnopis (angl. *ciphertext*). V našem primeru so sporočila podatki spletne aplikacije, ki jih želimo skriti pred zaledno storitvijo.

Kriptografija se deli v tri osnovne kategorije [1]:

- **Simetrična kriptografija**. Osebi se dogovorita o uporabi šifrirnega in dešifrirnega algoritma, za katerega poznata skupen ključ. Vsa kriptografija do leta 1976 je temeljila izključno na simetričnih metodah. Še danes se obsežno uporabljajo, predvsem pri algoritmih šifriranja podatkov in preverjanju integritete sporočila.

- **Asimetrična kriptografija**: Leta 1976 so Whitfield Diffie, Martin Hellman in Ralph Merkle svetu predstavili popolnoma drugačen način šifriranja podatkov. Pri asimetrični kriptografiji (pogosto imenovani tudi kriptografija z javnim ključem) ima uporabnik poleg zasebnega ključa tudi javni ključ. Asimetrična kriptografija je uporabljena pri digitalnih podpisih, reševanju problema varne izmenjave ključev ter

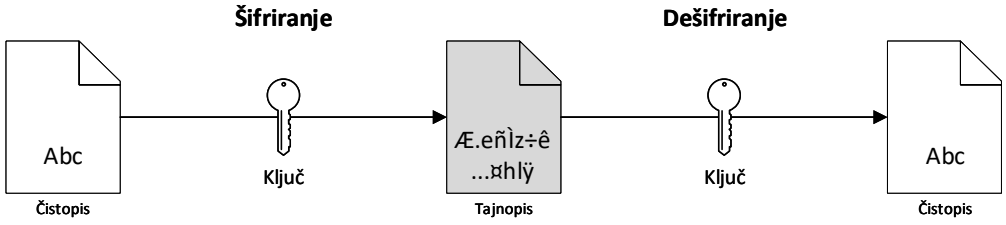
klasičnem šifriranju podatkov.

- **Kriptografski protokoli**. Ukvarjajo se z implementacijo kriptografskih algoritmov. Algoritme si lahko predstavljamo, kot gradnike, s katerimi so aplikacije, kot je na primer varna internetna komunikacija, realizirane.

2.1.1 Simetrična kriptografija

Simetrično kriptografijo je najlažje razumeti s klasičnim primerom o Ani in Bojanu. Kadar se želita pogovarjati po mediju, na katerem vso komunikacijo posluša tudi Oskar. Ana in Bojan ne želita, da bi kdorkoli drug lahko poslušal njun pogovor. Zato se najprej po varnem mediju dogovorita za ključ. Zdaj lahko Ana šifrira sporočilo, namenjeno Bojanu, na podlagi simetričnega algoritma. Algoritem kot vhodni podatek poleg sporočila uporabi dogovorjeni ključ. Bojan nato prejeti tajnopis dešifrira z enakim ključem. Primer pošiljanja sporočila je prikazan na sliki 2.1.

Trideset let je bil najpopularnejši DES (angl. *Data Encryption Standard*). Čeprav danes velja za šibek algoritem (njegov ključ je premajhen), šifriranje sporočila trikrat z DES še vedno velja za varen algoritem, imenovan 3DES (angl. *Triple DES*) [1].



Slika 2.1: Šifriranje in dešifriranje sporočila s simetrično kriptografijo

AES

AES (angl. *Advanced Encryption Standard*) je najbolj uporabljan simetrični

algoritem šifriranja [1]. Najdemo ga v protokolih IPsec (angl. *Internet Protocol Security*), SSL (angl. *Secure Sockets Layer*), SSH (angl. *Secure Shell*) in številnih drugih. Leta 2001 je NIST (angl. *National Institute of Standards and Technology*) označil šifrirni algoritem Rijndael kot novi AES. Rijndael sta razvila mlada belgijska kriptografa. Do danes ni znanih ranljivosti proti AES. Omogoča uporabo 128-, 192- ali 256-bitnih ključev. Trenutno obstaja osem odobrenih načinov delovanja algoritma: pet za zagotavljanje zaupnosti (ECB, CBC, CFB, OFB, CTR), eden za avtentikacijo (CMAC) in dva združena načina za zagotavljanje zaupnosti in avtentikacije (CCM, GCM).

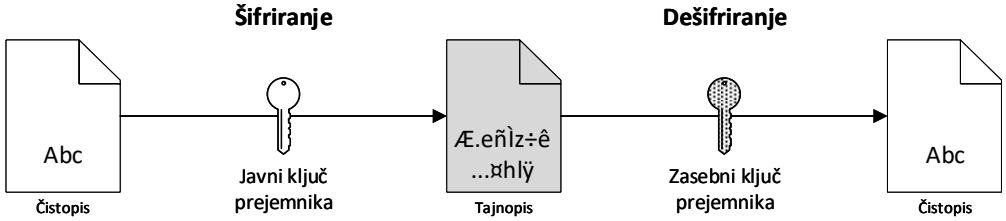
2.1.2 Asimetrična kriptografija

Pri simetrični kriptografiji je problem, kako varno izmenjati zasebni ključ. To težavo je mogoče odpraviti z asimetrično kriptografijo, ki temelji na konceptu zasebnega in javnega ključa. Če Ana želi poslati sporočilo na varen način Bojanu, mora najprej pridobiti njegov javni ključ. Javni ključ Bojana je lahko javno objavljen (na voljo vsem), saj se uporablja le za šifriranje, in ne dešifriranje sporočil. Ana nato pri šifriranju sporočila uporabi Bojanov javni ključ in tajnopis sporočila pošlje Bojanu. Ko ga Bojan prejme, uporabi svoj zasebni ključ za dešifriranje tajnopisa v prvotno sporočilo. Zasebni ključ mora Bojan skrbno varovati, saj brez njega prejetih sporočil ni mogoče dešifrirati. Primer uporabe prenosa sporočila je prikazan na sliki 2.2.

V primerjavi s simetrično je asimetrična kriptografija časovno potratnejša operacija, zato se običajno uporablja hibridni model, kjer je asimetrični algoritem namenjen le izmenjavi ključa. Za nadaljnjo komunikacijo pa uporabljamo simetrični algoritem.

V praksi se pogosto uporabljajo tri družine asimetrične kriptografije. Glede na njihov računski problem jih lahko razvrstimo v [1]:

- kriptosistem faktorizacije produkta,
- kriptosistem diskretnega logaritma in



Slika 2.2: Primer šifriranja in dešifriranja sporočila z asimetrično kriptografijo

- kriptosistem eliptične krivulje.

Kriptosistem diskretnega logaritma

V številnih kriptografskih algoritmih varnost temelji na težavnosti računanja diskretnih logaritmov. Znana primera sta Diffie-Hellmanova izmenjava ključa (angl. *Diffie–Hellman key exchange*) in ElGamalov kriptosistem [1].

Kriptosistem faktorizacije produkta

Najbolj znan in trenutno najbolj uporabljan asimetrični algoritem je RSA (Rivest-Shamir-Adleman). V osnovi RSA deluje na matematičnem problemu faktorizacije števil. Množenje dveh velikih praštevil je računsko lahka operacija, vendar je faktoriziranje rezultata za današnje računalnike še vedno težko rešljiv problem. Danes je RSA uporabljen v širokem naboru aplikacij. Najpogosteje ga zasledimo pri varni izmenjavi manjše količine podatkov (na primer pri izmenjavi ključev in digitalnih podpisov) [1].

Kriptosistem eliptične krivulje

ECC (angl. *Elliptic Curve Cryptography*) temelji na uporabi eliptičnih krivulj. V primerjavi z RSA potrebuje manjše ključe za delovanje in omogoča učinkovitejšo implementacijo, še vedno pa zagotavlja enako raven varnosti [2]. Zaradi majhnih sistemskih zahtev je ECC še posebej priljubljen v napravah IoT (angl. *Internet of things*). Ker uporablja majhne ključe, naprave ne potrebujejo veliko delovnega spomina za izvedbo šifrirnega algoritma [3].

2.1.3 Digitalni podpis

Digitalni podpisi so med najpomembnejšimi kriptografskimi orodji. Uporabljajo se za širok nabor aplikacij, od digitalnih potrdil za varno spletno nakupovanje do digitalnega podpisovanja dokumentov. Digitalni podpis zagotavlja, da je določena oseba določila vsebino. Digitalni podpisi za delovanje uporabljajo asimetrično kriptografijo. Če želi Ana sporočilo digitalno podpisati, za to uporabi svoj zasebni ključ. Bojan lahko nato preveri z Aninim javnim ključem, ali je sporočilo res podpisala ona [1].
V diplomskem delu je uporabljen algoritem ECDSA (angl. *Elliptic Curve Digital Signature Algorithm*) za digitalno podpisovanje. ECDSA v ozadju uporablja asimetrični algoritem ECC[1].

2.1.4 Varnost v spletnih aplikacijah

Če uporabnik želi uporabiti spletno aplikacijo, jo mora prenesti iz spletnega strežnika. To pomeni, da je kakršnakoli varnost, implementirana na ravni aplikacije, varna, dokler izvorna koda (JavaScript) aplikacije med prenosom ni bila spremenjena. Zato bi bilo zmotno misliti, da protokola SSL ne potrebujemo, če je zaščita implementirana v aplikaciji. Le s SSL smo lahko prepričani, da so prejete datoteke prave (take, kot nam jih pošlje spletni strežnik). To hkrati pomeni, da so spletni strežniki pogosto tarča napada. Tu uporabnik nima druge izbire kot zaupati strežniku, da mu bo zagotovil pravo spletno aplikacijo.

[1] S. Turner, D. Brown, *RFC 5753*. Dostopno na: https://tools.ietf.org/html/rfc5753#page-4. (pridobljeno 31. 5. 2017).

2.2 Okolje in zahteve

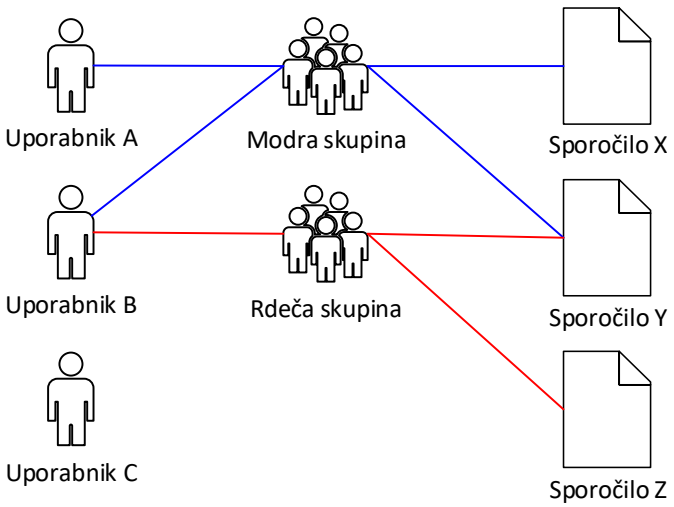
Podatke, ki jih spletne aplikacije potrebujejo za delovanje, običajno lahko razdelimo na tri podatkovne kategorije:

- uporabnike,
- skupine in
- podatke (na primer artikle, članke, dogodke ...).

Zato celotno delo temelji na okolju, v katerem spletna aplikacija upravlja z uporabniki, s skupinami in podatki. Skupine povezujejo uporabnike s podatki v smiselne sklope (na primer skupina za administratorje, spletne razvijalce, stranke podjetja ...). Uporabniki in podatki običajno pripadajo več skupinam hkrati. Pri tem je tudi implementirana vloga izmenjave podatkov z drugimi uporabniki. Če želimo deliti podatke z izbranimi uporabniki, izdelamo skupino, v katero so dodani kot člani. Skupini nato dodelimo dostop do podatkov.

Dodatna predpostavka za okolje je, da je uporabnikov v sistemu več, kot je skupin. Ta predpostavka je pomembna, ker je vplivala na zasnovo izmenjave podatkov in pri navedenih časovnih ocenah. Če ta pogoj ne velja, je implementacija še vedno enako varna, le časovna učinkovitost implementacije ni optimalna.

Opis povezovanja podatkovnih tipov
Imamo uporabnike A, B, in C, podatke X, Y in Z, modro skupino, h kateri spadajo uporabniki A in B ter podatki X in Y, rdečo skupino, h kateri spadajo uporabnik B ter podatki Y in Z. Uporabnik C ne pripada nobeni skupini in ne vidi nobenih podatkov. Opisani scenarij je prikazan na sliki 2.3. Enak opis je uporabljen kot primer pri drugih skicah in razlagah implementacije.



Slika 2.3: Primer scenarija

Pri implementaciji zaščite podatkov morajo biti zagotovljeni naslednji pogoji:

- **Varna hramba**
  Ključi in podatki morajo biti shranjeni v zalednem sistemu v tajnopisu. Zaledni sistem ne sme omogočati dešifriranja podatkov. Napadalec, ki vdre v zaledni sistem in ne pozna ključev, ne more razumeti vsebine podatkov. Tako so lahko vsi shranjeni podatki obravnavani kot varni [4].

- **Izmenjava in odvzem podatkov**
  Omogočena mora biti izmenjava šifriranih podatkov med uporabniki, ne da bi zaledna storitev, odgovorna za izmenjavo in hrambo, izmenjane podatke lahko dešifrirala. Samo uporabniki v skupini lahko berejo, spreminjajo ali brišejo podatke, ki spadajo k tej skupini.
  Če uporabnik zapusti skupino, morajo biti vsi ključi, znani skupini, zamenjani. Če je na primer uporabniku A odvzeto članstvo modre skupine, si s preteklimi ključi skupine ne more pomagati pri dešifriranju podatkov, tudi če bi mu bil omogočen fizični dostop do shrambe.

- **Integriteta (celovitost) podatkov**
  Za vse podatke mora obstajati način preverjanja, ali so podatki nedotaknjeni, kakršnokoli spreminjanje podatkov nepooblaščene osebe mora biti zaznano.

- **Preprosta uporaba**
  Za uporabnike mora biti celotno delovanje zaščite podatkov samodejno, brez dodatnih korakov. Kljub temu varnost ne sme biti ogrožena. Šifriranje in dešifriranje podatkov ne smeta občutno vplivati na uporabniško izkušnjo in odzivnost uporabniškega vmesnika.

2.3 Izmenjava podatkov

Implementacija izmenjave podatkov mora uporabnike razbremeniti misli o varnosti zaledne storitve. Edina naloga zaledja mora biti zagotavljanje podatkov spletni aplikaciji. Zato je za dosego varnosti shranjenih podatkov v naslednjih korakih opisana rešitev v spletni aplikaciji na strani odjemalca. Za izmenjavo podatkov morajo uporabniki poznati ustrezne ključe za dostop do podatkov, zato je predstavljen način izmenjave ključev med uporabniki. Razen javnih ključev morajo biti vsi ostali zaščiteni enako kot podatki, saj se uporabljajo za dostop do podatkov. Posledično mora biti način izmenjave ključev enako zaščiten kot podatki.

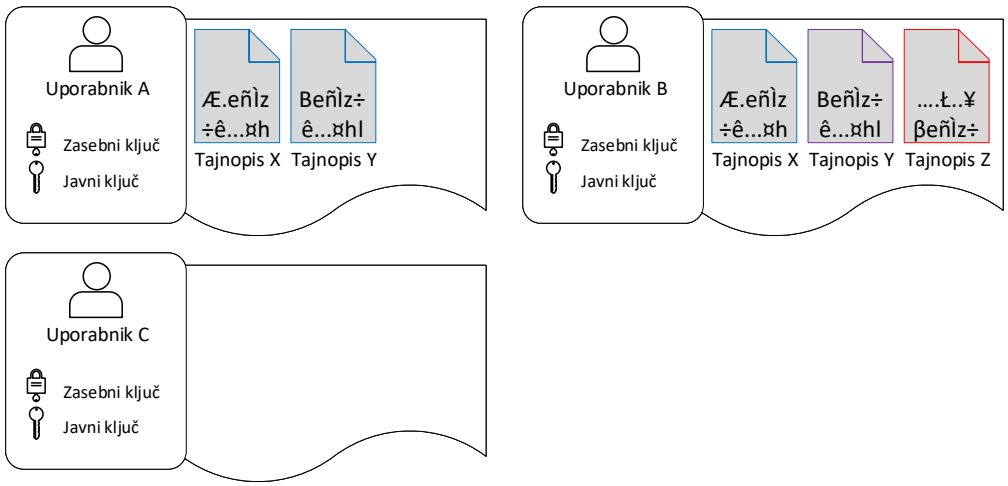
2.3.1 Korak 1: Začetna ideja

Vsak uporabnik aplikacije ima svoj javni in zasebni ključ. Oba sta shranjena v zalednem strežniku. Vendar pa je zasebni ključ že na strani aplikacije šifriran z osebnim geslom uporabnika (shranjen je v tajnopisu). Uporabniki za hrambo in deljenje podatkov uporabljajo asimetrično kriptografijo (podatke šifrirajo z javnim ključem in dešifrirajo z zasebnim).

Če uporabnik želi dati podatek v souporabo s skupino ali skupinami,

v katerih je skupaj n uporabnikov, mora narediti n kopij podatka. Vsaka kopija je namenjena enemu od uporabnikov skupine in je šifrirana z njegovim javnim ključem. Hkrati mora uporabnik (pošiljatelj) priložiti digitalni podpis podatka. Z digitalnim podpisom drugim uporabnikom dokaže, da je sporočilo res sestavil on in da ni bilo spremenjeno (enako kot pri protokolu S/MIME[2]). Tako lahko vsak uporabnik dešifrira kopijo, namenjeno njemu. Če nek uporabnik v skupini sporočilo spremeni, mora ponovno šifrirati n kopij (vsako kopijo s ključem, ki je bil dodeljen posameznemu uporabniku). Nadomestiti mora vse obstoječe kopije v zaledni shrambi.

Na sliki 2.4 je prikazano, kakšno bi bilo stanje v sistemu po scenariju s slike 2.3. Uporabniki A, B in C imajo poleg svojega javnega in zasebnega ključa namenski prostor za hrambo njim namenjenih podatkov. Ta prostor poimenujemo sef. Ker sta uporabnika A in B oba člana modre skupine, k modri skupini pa spada podatek X, imata oba v svojem sefu kopijo podatka X (šifriranega z njunim javnim ključem).



Slika 2.4: Korak 1 – stanje testnega scenarija

[2]B. Ramsdell, S. Turner, *RFC 5751*. Dostopno na: https://tools.ietf.org/html/rfc5751. (pridobljeno 20. 5. 2017).

Vlogo skupin (dodajanje in odvzemanje uporabnikov) upravlja zaledni strežnik, skupine pa ne nosijo nobene dodatne zaščite. So le abstraktna raven, implementirana na zalednem delu, kar pomeni, da je implementacija s stališča varnosti dokaj ranljiva. Aplikacijo je mogoče zlahka pretentati, da nek podatek da v souporabo napadalcu. Skupina v zalednem sistemu je preprosto predstavljena kot seznam uporabnikov, ki ga lahko vsak administrator spremeni.

Slaba učinkovitost

Prva slabost, ki jo lahko opazimo, je zelo slaba učinkovitost. Za vsakega uporabnika izdelamo svojo kopijo sporočila, ki jo je nato treba še šifrirati. Dodatno je uporaba asimetričnega šifriranja v primerjavi s simetričnim občutno počasnejša.

Potratna shramba

Ker za vsakega uporabnika v skupini hranimo kopijo podatkov, to povečuje zahteve po prostoru, ki ga potrebujemo za hrambo. Poleg neučinkovite hrambe moramo pri implementaciji zelo paziti, da ne bo prihajalo do nekonsistentnosti podatkov.

Časovne zahtevnosti na odjemalčevi strani

Legenda: n = število podatkov, dostopnih skupini, u = število članov skupine, m = število uporabnikov z dostopom do obravnavanega podatka.

- Dodajanje člana O(n)
  Pri dodajanju člana je treba šifrirati vse podatke (n), do katerih je skupini omogočen dostop.
- Odstranjevanje člana O(1)
  Ob odstranitvi člana mora le zaledna storitev odstraniti kopije podatkov, ki pripadajo članu. Potrebe po menjavi ključev ni.

- Dodajanje podatka O(u)
  Ob dodajanju podatka v skupino je treba vsem članom (u) šifrirati njihovo kopijo podatka.

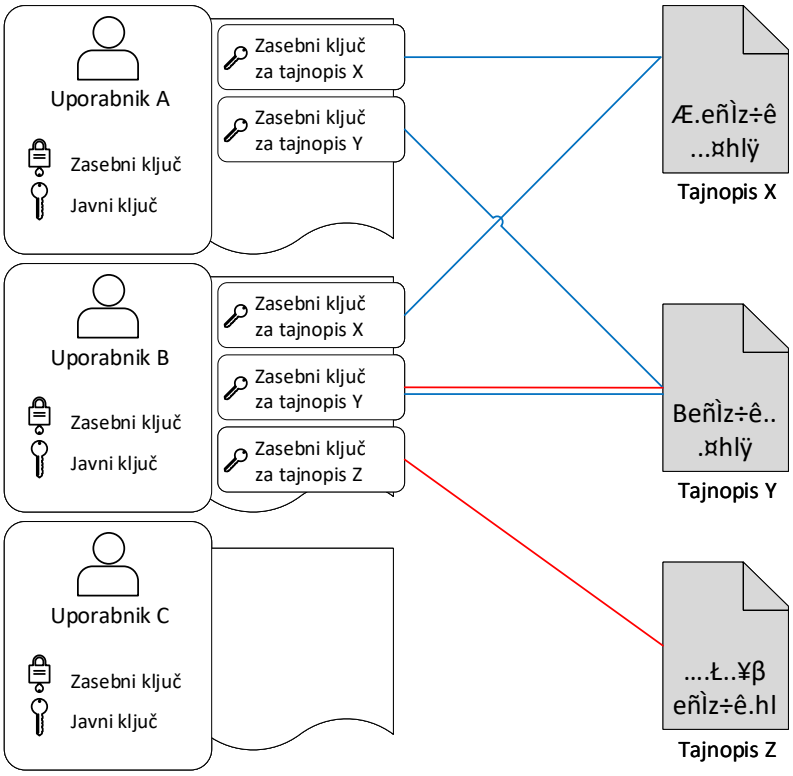
- Sprememba podatka O(m)
  Ob spremembi podatka je treba vsem uporabnikom (m), ki jim je omogočen dostop na podlagi skupin, posodobiti njihovo kopijo podatka.

2.3.2 Korak 2: Odprava kopij

Boljšo učinkovitost lahko dosežemo z odpravo kopij tajnopisov in uvedbo simetrične kriptografije. Vsak uporabnik še vedno ima svoj par ključev. Spremenimo le način šifriranja podatkov. Ko uporabnik želi dati podatek v souporabo s skupino, izdela nov zasebni ključ, s katerim šifrira podatek (simetrična kriptografija). Namesto izdelave kopij sporočila (kot je bilo izvedeno v prvem koraku), je treba izdelati le kopije zasebnega ključa podatka. Vsako kopijo ključa šifriramo z javnim ključem uporabnika. Na ta način smo odpravili kopije sporočil. Kar moramo storiti za vsakega uporabnika posebej, je le kopija zasebnega ključa podatka, s tem se občutno zmanjša potrebni prostor in časovno potratnost. Primer opisanega je prikazan na sliki 2.5. Opazimo lahko, da so v uporabnikovih sefih zdaj namesto tajnopisov podatkov, hranjeni tajnopisi zasebnih ključev za dešifriranje tajnopisov podatkov.

Časovne zahtevnosti na odjemalčevi strani
Upravljanje skupin je še vedno prepuščeno zalednemu strežniku, vendar si vseeno oglejmo izboljšave časovnih zahtevnosti v tem koraku.
Legenda: n = število podatkov, dostopnih skupini, u = število članov skupine.

- Dodajanje člana O(1)
  Pri dodajanju člana je treba s članom deliti le ključe podatkov.



Slika 2.5: Korak 2 – stanje testnega scenarija

- Odstranjevanje člana O(n)
  Ob odstranitvi člana je treba za vse podatke (n), za katere je član poznal ključ, izdelati novega, drugače bi lahko v primeru fizičnega dostopa še vedno dešifriral podatke.

- Dodajanje podatka O(u)
  Ob dodajanju podatka je treba vsem članom (u) v skupini poslati ključ.

- Sprememba podatka O(1)
  Sprememba podatka ne zahteva nobenih dodatnih operacij, ključ ostane enak.

2.3.3 Korak 3: Skupine kot aktivni del pri zaščiti

V tem koraku so skupine dodane kot aktivni člen pri zaščiti s ciljem varovanja zapisa skupine pred nepooblaščenimi spremembami. Hkrati je optimizirano delovanje.

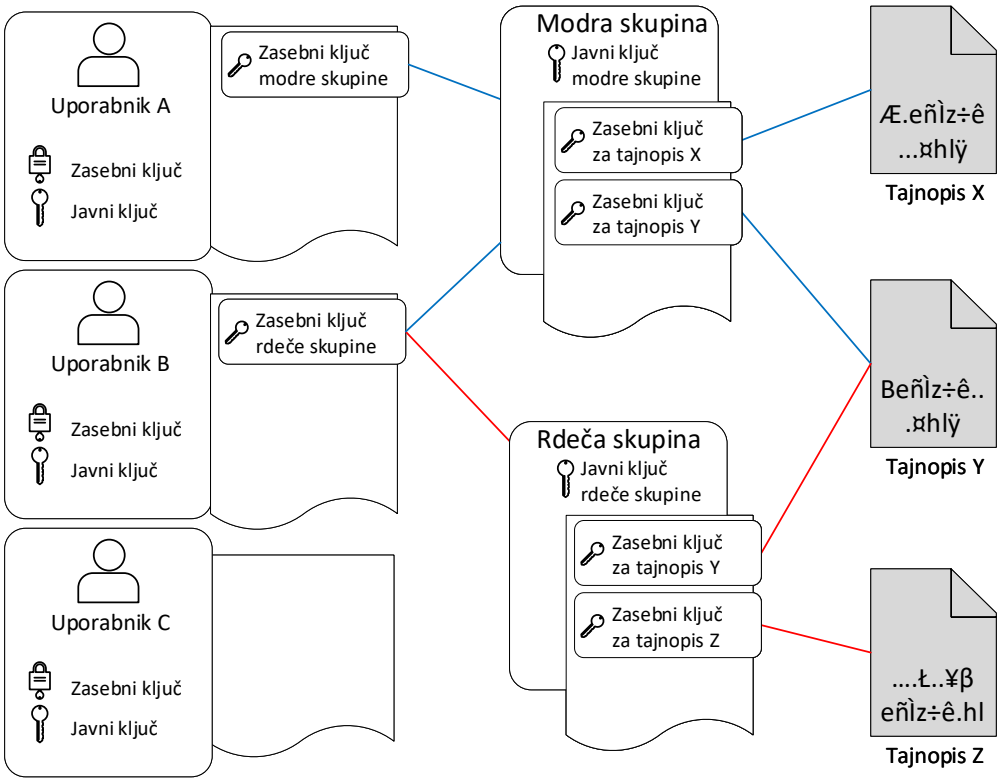
Vsaka skupina dobi svoj javni in zasebni ključ ter namenski prostor (sef) za hrambo vsebine, šifrirane z njenim javnim ključem, enako, kot imajo to uporabniki. Hrambo zasebnih ključev za podatke lahko zdaj premaknemo iz uporabnikovega sefa v sef skupin. Namesto da je ključ podatka šifriran z javnim ključem uporabnika, je zdaj šifriran z javnim ključem skupine. Zasebni ključ skupine pa je šifriran z javnim ključem uporabnika. S tem smo vlogo shrambe ključev podatkov prestavili z uporabnikov na skupine. Primer stanja s skupinami je prikazan na sliki 2.6. Če prikaz primerjamo s sliko iz drugega koraka (2.5), opazimo dodano vmesno raven skupin.

Dodatno vpeljani mehanizem je digitalno podpisovanje seznama članov skupine z zasebnim ključem skupine. S tem se zaščitimo, da bi zunanja oseba z dostopom do zbirke podatkov spremenila seznam uporabnikov skupine. Tudi če spremeni seznam članov, ga ne more digitalno podpisati, ker ne pozna zasebnega ključa skupine. Posledično lahko spletna aplikacija kakršnokoli spremembo v skupini zazna.

Časovne zahtevnosti na odjemalčevi strani

Legenda: n = število podatkov, dostopnih skupini, u = število članov skupine.

- Dodajanje člana O(1)
  Pri dodajanju člana je treba le šifrirati zasebni ključ skupine z javnim ključem uporabnika.
- Odstranjevanje člana O(n)
  Ob odstranitvi člana je treba za vse podatke (n) izdelati nove ključe.



Slika 2.6: Korak 3 – stanje testnega scenarija

- Dodajanje podatka O(1)
  Ob dodajanju podatka je treba le shraniti ključ v sef skupine.

- Sprememba podatka O(1)
  Sprememba podatka ne zahteva nobenih dodatnih operacij, ključ ostane enak.

Opazimo lahko, da je kot potratna operacija ostalo le odstranjevanja člana iz skupine. Predpostavimo lahko, da se ta v primerjavi z drugimi običajno izvede bolj poredko. Zato lahko rečemo, da je rešitev na odjemalčevi strani kar se da neopazna pri porabi sistemskih virov.

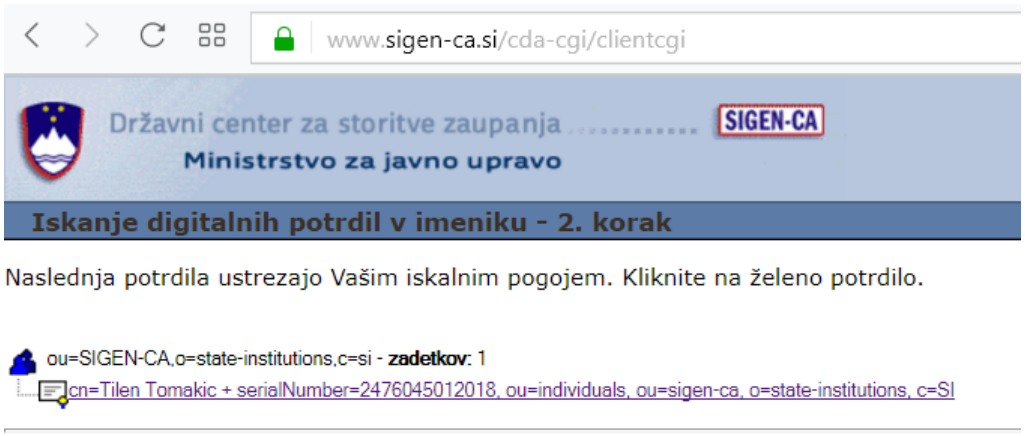
2.4 Infrastruktura in ranljivost

Implementacija omogoča, da je večji del zalednega sistema lahko ogrožen. Še vedno je pomembno, da sta del zalednega sistema, kjer hranimo javne ključe, in strežnik, ki omogoča prenos spletne aplikacije k odjemalcu, ustrezno zaščitena. Če se vsiljivcu uspe prebiti v enega izmed omenjenih delov, lahko prelisiči (šifriranje z lažnimi javnim ključem) ali zaobide varnostne mehanizme.

Sistem za hranjenje in zagotavljanje javnih ključev
Celotna implementacija je zasnovana na predpostavki, da zaupamo storitvi, ki zagotavlja javne ključe uporabnikov in skupin (zaupamo, da so ključi pravi). Če napadalec zamenja javni ključ nekega uporabnika ali skupine, tega spletna aplikacija ne more zaznati oz. ne more prepoznati, ali je javni ključ pravi ali ne. To lahko privede do potencialno neželene izmenjave podatkov s tretjo osebo. Da je nekaj narobe, lahko zazna le spletna aplikacija prizadetega uporabnika ali člana skupine, ko preveri ujemanje javnega in zasebnega ključa. Tu je pomembno poudariti, da tovrsten napad zahteva čas, takojšnja odtujitev podatkov ni mogoča, saj mora najprej nekdo od uporabnikov aplikacije sprožiti operacijo dodajanja ali brisanja.

Najbolje je sistem za zagotavljanje javnih ključev ločiti v novo zaledno storitev. Tako zmanjšamo kompleksnost storitve in lažje jamčimo višjo raven varnosti, saj je bistveno lažje izvesti varnostni pregled implementacije zaščite manjšega dela sistema kot celote.

Namesto da sami razvijemo in zaščitimo omenjeni del, lahko uporabimo overitelja digitalnih potrdil za zagotavljanje javnih ključev. Žal taka rešitev od uporabnika spletne aplikacije zahteva dodatne korake, saj spletni brskalniki v večini ne podpirajo digitalnega podpisovanja s shranjenimi



Slika 2.7: Poizvedba po javnem ključu na portalu SIGEN-CA

ključi v operacijskem sistemu uporabnika. Izjemi sta Mozilla Firefox[3] in Internet Explorer (po komponenti CAPICOM[4]).
Primer poizvedbe po javnem ključu na portalu SIGEN-CA je prikazan na sliki 2.7.

Nalaganje spletne aplikacije

Ker gre za spletno aplikacijo, moramo zaupati spletnemu strežniku, da nam zagotavlja pravo vsebino. V primeru vdora v strežnik lahko napadalec spremeni datoteke JavaScript spletne aplikacije ter s tem njeno delovanje. To pomeni, da lahko zaobide vse varnostne mehanizme in pridobi dostop do tajnopisov ter njihovih dešifrirnih ključev.

Najboljši scenarij bi bil, da aplikacije ne bi prenašali iz spletnega strežnika, temveč bi jo naložili lokalno iz uporabnikove naprave. To bi lahko

[3]Mozilla Corporation, *JavaScript crypto*. Dostopno na: https://developer.mozilla.org/en-US/docs/Archive/Mozilla/JavaScript_crypto. (pridobljeno 23. 6. 2017).

[4]Microsoft Corporation, *About Cryptography*. Dostopno na: https://msdn.microsoft.com/en-us/library/windows/desktop/aa375754(v=vs.85).aspx. (pridobljeno 23. 6. 2017).

dosegli tako, da bi spletno aplikacijo ponudili uporabnikom kot izvršljiv program. Celotna koda aplikacije bi bila v uporabnikovi napravi (ni potrebe po prenosu). Obstaja več orodij, ki spletno aplikacijo zapakirajo v izvršljivo datoteko, na primer Electron, NW.js, Cordova, Ionic.

Poglavje 3

Spletna aplikacija KalPass

Namen razvoja spletne aplikacije KalPass je bil na praktičnem primeru predstaviti implementacijo varne izmenjave podatkov med uporabniki spletne aplikacije. Opisani so ključni deli aplikacije in njihov razvoj ter delovanje predlaganih pristopov.

Spletna aplikacija KalPass je namenjena upravljanju gesel (prijavnih podatkov) podjetja. Vsakemu zaposlenemu v podjetju je z njegovim uporabniškim računom omogočen dostop do odobrenih gesel. Dostop do gesel je dodeljen na podlagi skupin, h katerim spada uporabnik. Upravljajo jih lahko le uporabniki, ki so v vlogi skrbnika.

Zaledni del, ki ga aplikacija uporablja, posebej ni predstavljen, saj vlogo zaščite podatkov v celoti prevzema spletna aplikacija. Zaledni del je večinoma le v vlogi shrambe, hkrati pa ne implementira nobenega ključnega mehanizma, predstavljenega v drugem poglavju.

Na začetku so predstavljene uporabljene tehnologije in razlogi za njihovo uporabo. Nato sledi predstavitev infrastrukture aplikacije in zaledja. V jedru sta predstavljena razvoj ključnih storitev in komponent Angular ter njihovo delovanje. Na koncu so izpostavljene varnostne pomanjkljivosti.

3.1 Tehnologije

Aplikacija KalPass je zasnovana na tehnologiji SPA. Za predstavljeno rešitev zaščite v drugem poglavju je to idealen način izdelave spletnih aplikacij. S stališča implementacije to pomeni, da lahko stanje aplikacije (podatke, ključe) hranimo v spominu naprave za celotni čas uporabe, ne da bi morali zaledno storitev ponovno povpraševati po njih. S tem je tudi zaledna storitev razbremenjena, saj mora ponujati le REST API z osnovnimi operacijami CRUD (angl. *Create, Read, Update and Delete*) za hrambo podatkov (uporabnikov, skupin in gesel).

3.1.1 Angular

Angular 2 je izšel septembra 2016, vendar ga ne smemo povezovati s predhodnikom AngularJS, saj je Angular 2 čisto novo orodje. Vse nadaljnje različice orodja Angular izhajajo iz prejšnjih različic in implementacije ne bodo drastično spreminjale (so polno združljive s prejšnjo različico). Če primerjamo prehod z orodja AngularJS na Angular 2, ugotovimo, da orodji medsebojno nista združljivi. Aplikacija KalPass je zgrajena na osnovi orodja Angular 4. Od zdaj naprej Angular 4 navajamo kot Angular [5].

Angular je strukturno orodje, namenjeno izgradnji aplikacij SPA. Aplikacija SPA je le HTML-dokument, ki se naloži ob odprtju spletne aplikacije. Ta dokument naloži ogrodje, ki nato dinamično menjava vsebino spletne strani. Tipično se pri aplikacijah SPA večina kode izvaja v spletnem brskalniku, zaledna storitev le zagotavlja podatke (REST API). Prednost pri tem je zmanjšana obremenjenost zalednega strežnika (prenesli smo vlogo izvajanja logike z zaledne storitve na odjemalčevo stran). Na ta način je odzivnost aplikacije neodvisna od obremenjenosti zaledne storitve s stališča delovanja uporabniškega vmesnika [6].

Pri izgradnji aplikacije KalPass je bil Angular uporabljen zaradi omogočanja hitrega in kakovostnega razvoja aplikacije. Razvijalcu ponuja velik nabor funkcionalnosti (lažje delo z obrazci, zagotavljanje podatkov komponentam, zaščito proti XSS-napadom (angl. *Cross-site Scripting*) ...). Hkrati pa je dobro strukturno orodje, saj razvijalca usmeri v vnaprej določen način strukturiranja aplikacije.

Ključni koncepti v orodju Angular so:

- moduli,
- komponente,
- storitve in
- poti.

Moduli

Aplikacije Angular so sestavljene iz enega ali več modulov. Vsaka aplikacija ima glavni modul, imenovan *AppModule*. To je lahko tudi edini modul v celotni aplikaciji (v manjših aplikacijah), vendar večina aplikacij obsega več modulov. Tipično modul vsebuje dele aplikacije, ki skupaj predstavljajo specifično funkcionalnost.

Komponente

Komponenta predstavlja določeno območje strani, imenovano pogled (angl. *View*). To je lahko navigacija, seznam uporabnikov, obrazec za urejanje gesla. Celotna aplikacija je le skupek komponent. Komponenta je sestavljena iz razreda in predloge, na podlagi katere Angular sestavi pogled. Logika komponente je definirana v razredu. Razred nato upravlja s pogledom na podlagi lastnosti in metod.

Storitve

Skoraj karkoli je lahko storitev. Običajno je storitev sestavljena iz sklopa spremenljivk in funkcij, ki aplikaciji zagotavljajo neko ozko namenjeno funkcionalnost. Komponente običajno uporabljajo eno ali več storitev za

pridobitev funkcionalnosti, ki jih potrebujejo.

Poti

Poti omogočajo navigacijo po aplikaciji od ene komponente do druge. V orodju Angular so podobne mehanizmu HTML-povezav, ki uporabniku omogočajo navigacijo med spletnimi stranmi. Glavna razlika je v tem, da se poti obravnavajo znotraj SPA, torej stran ni ponovno osvežena.

3.1.2 TypeScript

TypeScript[1] je odprtokodni jezik, ki ga je razvil Microsoft. Izdelan je bil z namenom lažjega in zanesljivejšega razvoja spletnih aplikacij. Je statično napisan jezik, kar pomeni, da je tip spremenljivk znan ob prevajanju jezika. TypeScript je le nadjezik jezika JavaScript (celotna koda JavaScript je hkrati veljavna koda TypeScript, ki je prevedena v JavaScript). Ker TypeScript omogoča definiranje podatkovnih tipov in ga je treba pred zagonom prevesti, to omogoča odkrivanje napak, še preden aplikacijo zaženemo [7].

3.1.3 SJCL

SJCL[2] (*The Stanford Javascript Crypto Library*) je kriptografska knjižnica JavaScript, ki jo je razvil laboratorij za računalniško varnost v Stanfordu. Ponuja simetrične in asimetrične algoritme, digitalno podpisovanje in zgoščevalne funkcije. V aplikaciji KalPass je za vse kriptografske algoritme uporabljena le knjižnica SJCL. Za uporabo te namesto drugih knjižnic sem se odločil, ker je podprta z raziskovalnim člankom[3], v primerjavi z drugimi je ena izmed starejših in pogosto uporabljanih ter deluje v vseh spletnih

[1]Uradna spletna stran TypeScript, https://www.typescriptlang.org. (pridobljeno 30. 5. 2017).

[2]Uradna spletna stran SJCL, https://crypto.stanford.edu/sjcl/. (pridobljeno 9. 6. 2017).

[3]E. Stark, M. Hamburg, D. Boneh. *Symmetric Cryptography in Javascript.* http://bitwiseshiftleft.github.io/sjcl/acsac.pdf. (pridobljeno 10. 6. 2017).

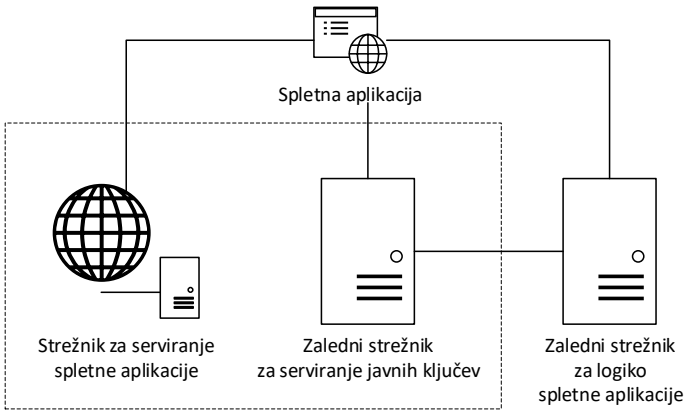
brskalnikih.

V aplikaciji je uporabljen simetrični algoritem AES z 256-bitnim ključem v načinu CCM[4]. Za asimetrični algoritem je vzet algoritem ECC s krivuljo P-256[5], pri digitalnih podpisih pa je uporabljen algoritem ECDSA.

3.2 Infrastruktura

Aplikacija KalPass uporablja v ozadju tri zaledne storitve, kot so prikazane na sliki 3.1:

- spletni strežnik za zagotavljanje spletne aplikacije,
- zaledno storitev za zagotavljanje javnih ključev uporabnikov in skupin,
- zaledno storitev za zagotavljanje vseh drugih funkcionalnosti, ki jih potrebuje KalPass.



Slika 3.1: Infrastruktura

[4]M. Dworkin. *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, maj 2004, http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38c.pdf. (pridobljeno 12. 6. 2017).

[5]D. McGrew, K. Igoe, M. Salter, *RFC 6090.* Dostopno na: https://tools.ietf.org/html/rfc6090#section-3.3. (pridobljeno 12. 6. 2017).

Spletni strežnik in zaledna storitev za zagotavljanje javnih ključev sta namerno ločena od drugih funkcionalnosti zaradi lažje zagotovitve ustrezne zaščite pred vdori. Za vso komunikacijo med aplikacijo in zaledjem ter zalednimi storitvami med seboj je uporabljen protokol HTTPS[6].

3.3 Razvoj

Aplikacijo sestavljajo trije osnovni sklopi:

- uporabniki,
- skupine in
- gesla.

Komponente spletne aplikacije vedno le sprožijo želeno akcijo ter na podlagi obljub[7] (angl. *promise*) čakajo na status akcije z namenom prikaza povratne informacije uporabniku, storitev pa poskrbi za izvedbo akcije.

V aplikaciji KalPass obstajajo tri storitve (kot je prikazano na sliki 3.2):

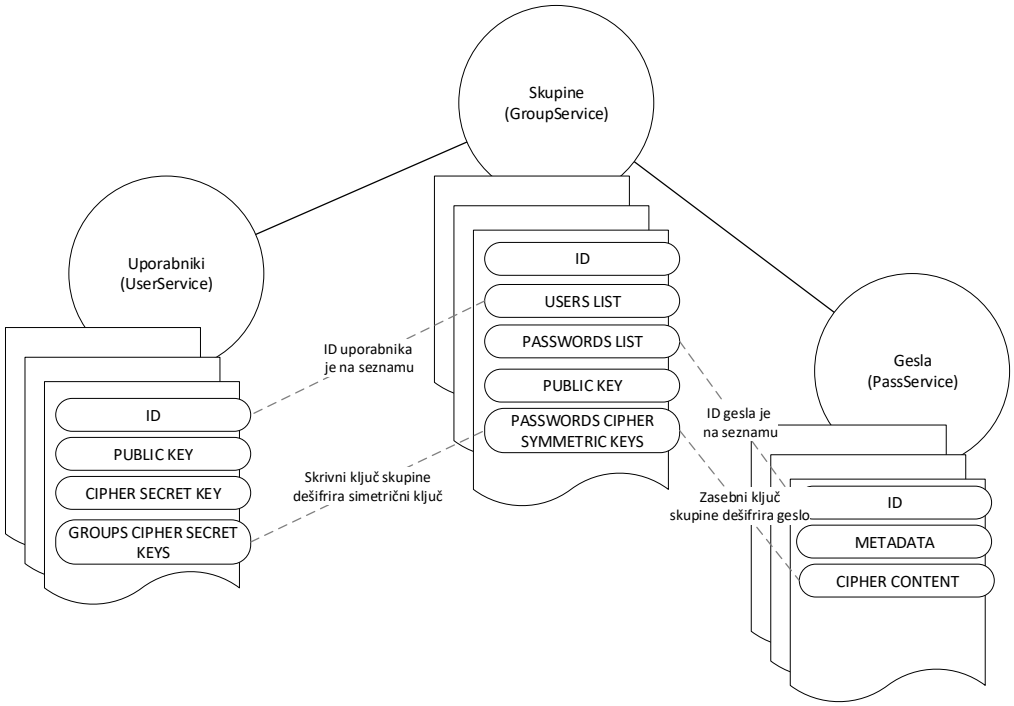
- storitev za upravljanje z uporabniki,
- storitev za upravljanje s skupinami ter
- storitev za upravljanje z gesli.

Vse storitve ponujajo naslednje funkcionalnosti za delo z entitetami (entitete so v tem kontekstu uporabniki, skupine ali gesla, odvisno od storitve):

- seznam entitet,
- prenos entitete iz zaledne storitve,

[6]E. Rescorla, *RFC 2818*. Dostopno na: https://tools.ietf.org/html/rfc2818. (pridobljeno 15. 6. 2017).

[7]Ecma International, *ECMA-262*, str. 483, pogl. 25.4. Promise Objects, http://www.ecma-international.org/ecma-262/6.0/ECMA-262.pdf. (pridobljeno 30. 6. 2017).



Slika 3.2: Relacija med storitvami in podatkovnimi tipi

- shranjevanje entitet v zaledno storitev,
- možnost naročnine o spremembi entitet (namenjeno za druge storitve in komponente),
- implementirana sta vmesnika Angular *Resolve*[8] in *CanActivate*[9]. Vmesnik *Resolve* uporabljajo komponente, ki za delovanje potrebujejo točno določeno entiteto (na primer prikaz podrobnosti uporabnika). Vmesnik *CanActivate* komponentam omogoča, da počakajo s svojo inicializacijo, dokler storitvi ni na voljo celoten seznam entitet. Komponenta za prikaz seznama uporabnikov je na primer v stanju čakanja, dokler storitev za uporabnike ne prenese vseh uporabnikov.

[8] *Dokumentacija za vmesnik Resolve, https://angular.io/api/router/Resolve.*
[9] *Dokumentacija za vmesnik CanActivate, https://angular.io/api/router/CanActivate.*

3.3.1 Avtentikacija

Avtentikacijska storitev

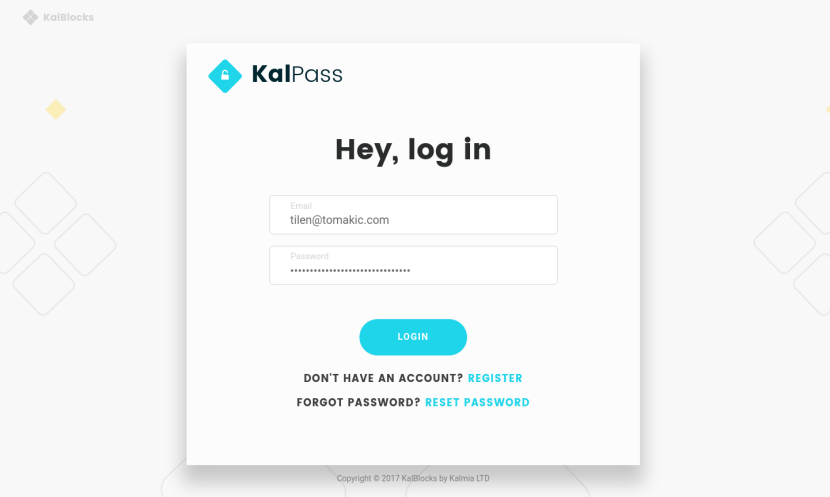
Glavna funkcija te storitve je prijava uporabnika v aplikacijo. Storitev tudi hrani vse informacije o trenutno prijavljenem uporabniku. To obsega žeton JWT (angl. *JSON Web Token*)[10] za avtentikacijo, zasebni ključ uporabnika in vse zasebne ključe skupin, katerih član je uporabnik.

Postopek prijave

Uporabnik potrebuje za prijavo le svoj e-naslov in osebno geslo (slika 3.3). Ker geslo uporabljamo tudi za dešifriranje zasebnega ključa, ga ne moremo poslati neposredno zaledni storitvi, saj bi to izničilo vso varnost oz., povedano drugače, bi zalednemu sistemu omogočili dešifriranje vsebine v zbirki podatkov. To preprečimo tako, da namesto gesla pošljemo rezultat zgoščevalne funkcije, ki kot vhod prejme geslo. Na ta način zaledni strežnik ne more priti do izvorne vrednosti gesla, ki bi jo lahko uporabil za dešifriranje. Prav tako pa nismo glede varnosti nič prikrajšani, saj zaledna storitev na svoji strani tako ali tako uporabi zgoščevalno funkcijo za primerjavo gesla v zbirki podatkov. Tu smo le prenesli vlogo, kdo požene zgoščevalno funkcijo nad geslom.

Ko zaledna storitev avtenticira uporabnika na podlagi ujemanja zgoščevalne funkcije gesla in e-poštnega sporočila, mu pošlje žeton JWT, tajnopis njegovega zasebnega ključa, njegov javni ključ ter tajnopis vseh zasebnih ključev skupin, h katerim pripada. Nato spletna aplikacija uporabi izvorno geslo uporabnika za dešifriranje zasebnega ključa. Za dodatno varnost storitev preveri, ali se zasebni ključ uporabnika ujema z njegovim javnim ključem. Če se ne ujema, pomeni, da je prišlo do napada (nekdo je spremenil

[10] M. Jones and J. Bradley and N. Sakimura, *RFC 7519*. Dostopno na: https://tools.ietf.org/html/rfc7519. (pridobljeno 14. 7. 2017).



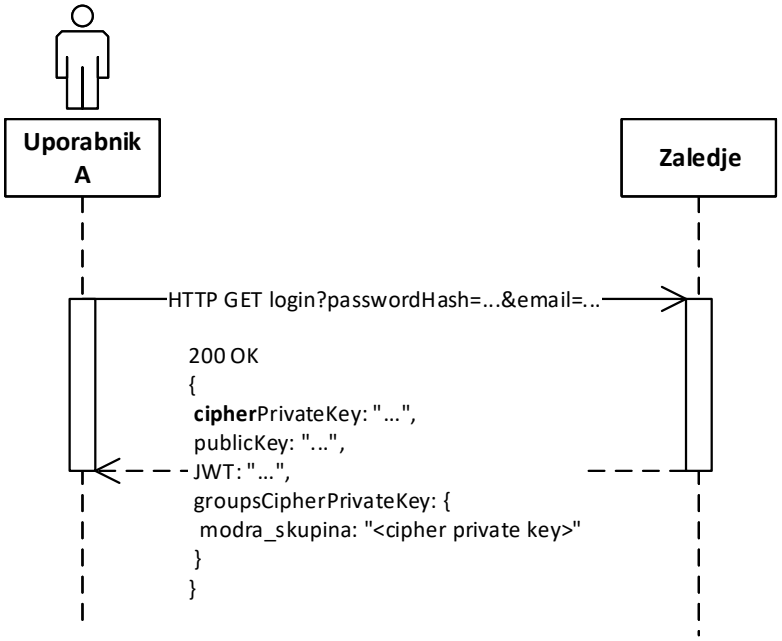
Slika 3.3: Prijavni obrazec

javni ključ uporabnika). Zasebni ključ je shranjen v sejni shrambi[11] (angl. *session storage*). Sejna shramba je bila uporabljena z namenom, da vsebino ohrani, tudi če je spletna stran osvežena, hkrati pa zagotavlja dobro zaščito, saj ima vsako okno spletnega brskalnika svojo sejno shrambo. Ko uporabnik zapre spletno aplikacijo ali odpre drugo spletno stran, se sejna shramba izbriše. Potek medmrežne komunikacije prijave je predstavljen na sliki 3.4.

Menjava gesla

Če uporabnik pozna svoje trenutno geslo, zanj ne obstaja noben dodatni korak. Ob postopku menjave gesla storitev hkrati šifrira zasebni ključ uporabnika z novim geslom. Če je uporabnik geslo pozabil in začne postopek menjave, bo moral zamenjati javni in zasebni ključ. To pomeni, da izgubi članstvo vseh skupin. Uporabnik bo moral skrbnika prositi, da ga znova doda v skupine.

[11] W3C, *Web Storage (druga izdaja)*, https://www.w3.org/TR/webstorage/#the-sessionstorage-attribute. (pridobljeno 30. 6. 2017).



Slika 3.4: Sekvenčni diagram medmrežne komunikacije pri prijavi

3.3.2 Skupine

V sistemu vedno obstaja posebna skupina, imenovana *admins*. Uporabniki te skupine so v vlogi skrbnikov. Samo skrbniki lahko izdelujejo, urejajo ter brišejo skupine in uporabnike. Običajen uporabnik ne more sam od sebe zapustiti skupine, saj ključev ni mogoče zamenjati, ne da bi zanje vedel uporabnik, ki zapušča skupino ali pa zaledni sistem. Zato lahko člana iz skupine odstrani le drugi član v skupini (v našem primeru skrbnik).

Storitev za upravljanje s skupinami

Za zagotavljanje informacij o uporabnikih in njihovo upravljanje je zadolžena storitev, imenovana *GroupService*.

Podatkovni tip skupine je predstavljen z naslednjimi atributi (glejte sliko 3.2):

- ID skupine,
- ime skupine,
- javni ključ skupine,
- seznam šifriranih zasebnih ključev gesel z javnim ključem te skupine,
- seznam članov skupine (seznam ID uporabnikov),
- seznam gesel, ki jih imajo člani pravico dešifrirati (seznam ID gesel),
- digitalni podpis.

Vloga digitalnega podpisa je overjanje, da ključni metapodatki skupine niso bili spremenjeni (med hrambo v zbirki podatkov). Če se digitalni podpis ne ujema, storitev onemogoči spreminjanje skupine in prikaže obvestilo uporabniku o potencialnem napadu. Digitalni podpis je izveden nad metapodatkoma seznama članov skupine in imena skupine. Pri digitalnem podpisovanju je uporabljen zasebni ključ skupine.

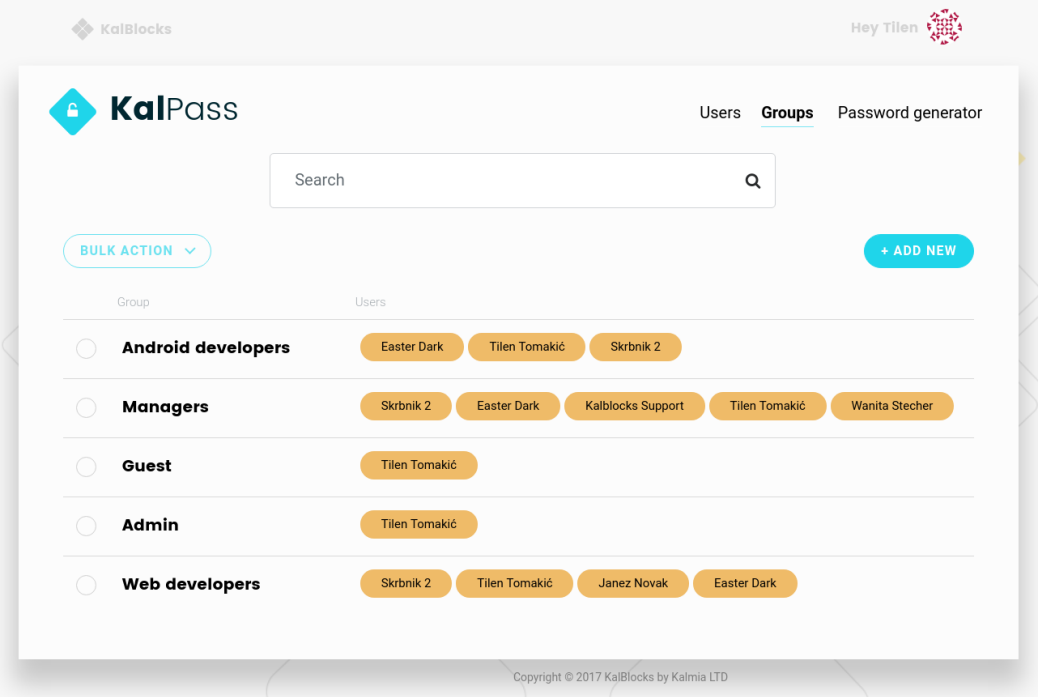
Pregled skupin

Do komponent za pregled skupin (slika 3.5) in dodajanje/urejanje skupine lahko pride le uporabnik, ki je v vlogi skrbnika.

Izdelava skupine

Že ob odprtju obrazca za izdelavo nove skupine (slika 3.6) je skrbnik samodejno dodan kot njen član. Odstranitev ustanovitelja skupine ni mogoča, saj spletna aplikacija ustanovitelja izdela zasebni in javni ključ skupine, torej mora biti obvezno v njej. Če uporabnik ni del skupine, pomeni, da ne pozna njenih skrivnosti, kar v tem primeru ne bi veljalo.

Ko skrbnik sproži shranjevanje, komponenta le preveri veljavnost vnesenih podatkov, nato pa postopek shranjevanja prepusti storitvi *GroupService*. Ta izdela nov zasebni in javni ključ skupine. Digitalno podpiše objekt skupine. Izdelan zasebni ključ zašifrira za vsakega člana skupine z njegovim



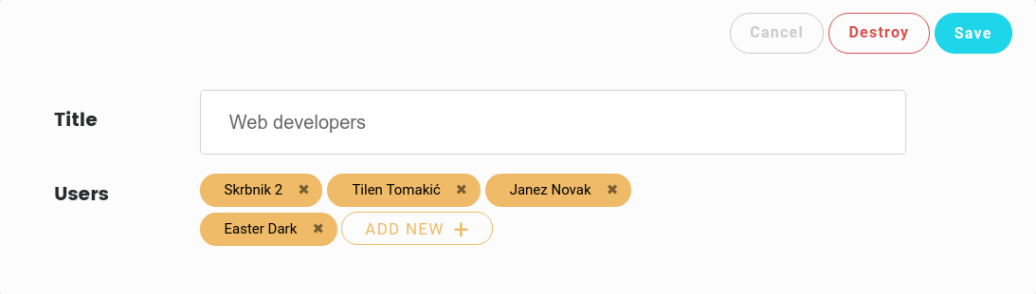
Slika 3.5: Seznam skupin

javnim ključem. Nato celoten objekt skupine skupaj s tajnopisi zasebnega ključa za člane pošlje zaledni storitvi.

Spreminjanje skupine

Ko skrbnik spremeni skupino, se zgodi podoben postopek kot ob izdelavi skupine. Vedno se poleg spremembe objekta skupine na novo izdela digitalni podpis skupine. Če je bil spremenjen seznam članov skupine, pa se postopek za dodajanje ali odstranjevanje člana razlikuje.

Dodajanje člana: Ko skrbnik doda uporabnika na seznam članov skupine, *GroupService* zašifrira zasebni ključ skupine z javnim ključem uporabnika ter pošlje spremenjen objekt skupine in tajnopis ključa zaledni storitvi. Na novo dodan član ima zdaj dostop do vseh gesel, ki pripadajo tej skupini. Potek dodajanja člana je razviden s slike 3.7.



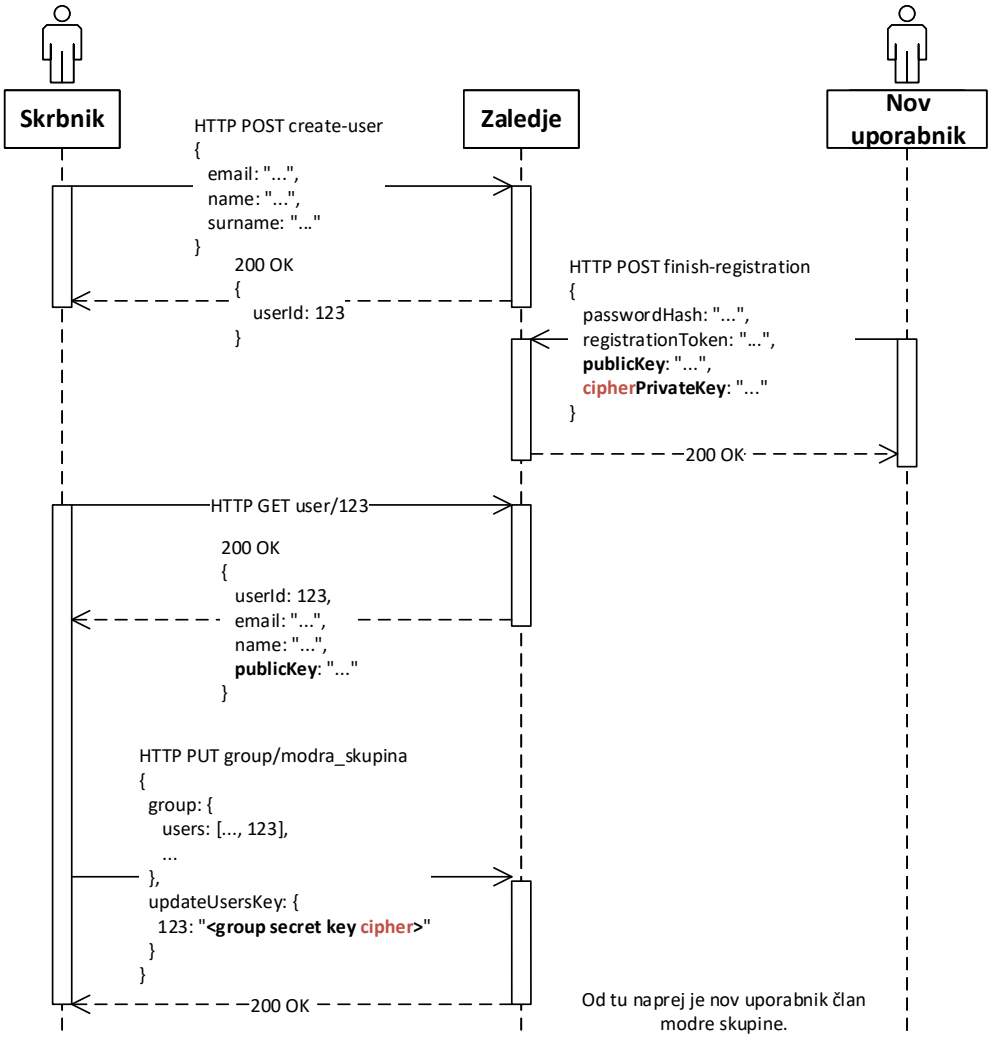
Slika 3.6: Komponenta za izdelavo/urejanje skupine

Odstranjevanje člana: Najprej je treba poudariti, da skrbnik sebe ne more odstraniti iz skupine. To varovalo je implementirano iz dveh razlogov:

- Če vsi skrbniki zapustijo skupino, efektivno izgubimo dostop do gesel, ki so bila dodeljena le tej skupini.
- Skrbnik sebe ne more odstraniti, saj bi še vedno poznal ključe, kar ni v skladu z zahtevo, da uporabniki, ki niso del skupin, ne poznajo ključev skupine.

Ker odstranjeni član pozna zasebni ključ skupine in zasebne ključe gesel, ki so zaupani tej skupini, moramo vse te ključe zamenjati. Veljati mora, da uporabniki, ki niso člani skupine, ne poznajo ključev skupine, saj nočemo, da bi imel odstranjeni član možnost dešifriranja v primeru, da mu je omogočen dostop do zbirke podatkov.

Zato ob odstranitvi člana *GroupService* izdela nov zasebni in javni ključ. Z novim zasebnim ključem ponovno šifrira ključe vseh gesel. Hkrati pa enako kot pri izdelavi skupine vsem članom šifrira zasebni ključ skupine z njihovim javnim ključem.



Slika 3.7: Sekvenčni diagram izdelave novega uporabnika in dodajanja tega v skupino

Brisanje skupine

Izbris skupine je mogoč le ob pogoju, da z njo niso v souporabi nobena gesla. V nasprotnem primeru mora skrbnik najprej poskrbeti za njihovo odstranitev. Problem pri odstranitvi ostaja podoben kot pri odstranjevanju člana iz skupine. Ne smemo dovoliti, da uporabniki, ki jim ni omogočen dostop do gesel, poznajo ključe za njihovo dešifriranje. S pogojem, da s skupino niso v souporabi nobena gesla, pa zagotovimo, da bo uporabnik najprej moral odstraniti souporabo, s tem pa sprožil ustrezne postopke za ponovno šifriranje ključev gesel.

Tabela 3.1 povzema operacije za opisana dejanja nad skupinami.

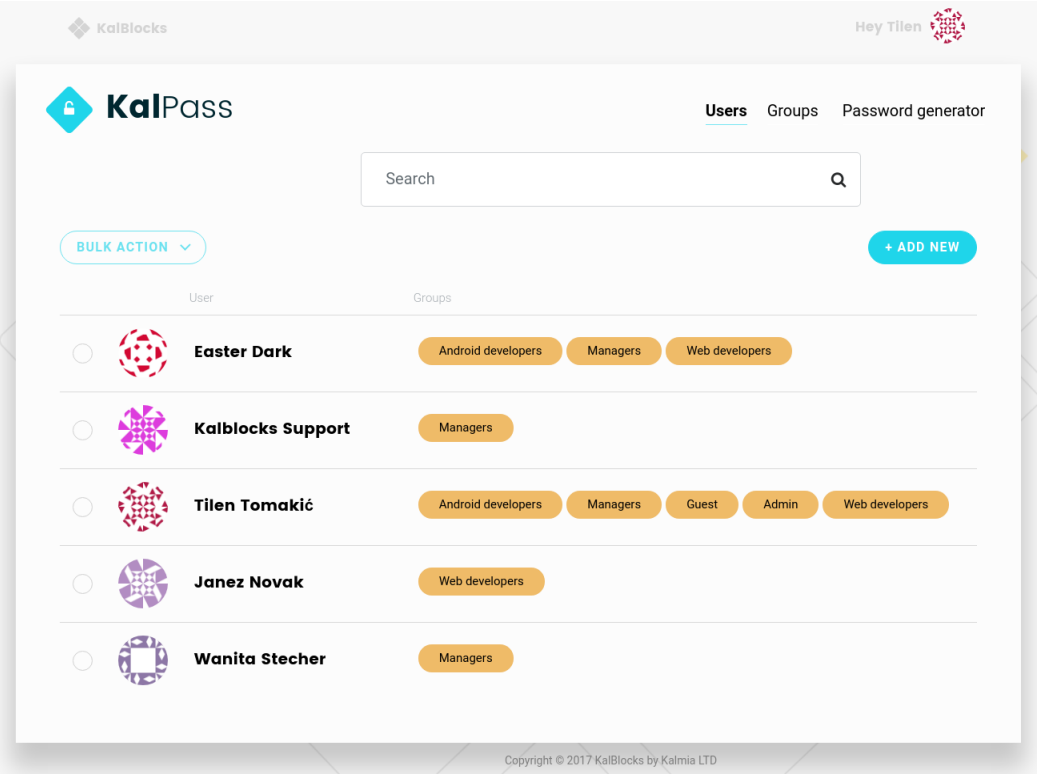
dejanje	**operacije**
Izdelava skupine	1. Izdelaj javni in zasebni ključ. 2. Izdelaj objekt skupine s seznamom članov. 3. Digitalno podpiši objekt skupine. 4. Shrani objekt skupine in deli tajnopis zasebnega ključa s člani.
Sprememba skupine – dodan član	1. Dodaj člana v objekt skupine. 2. Digitalno podpiši objekt skupine. 3. Shrani objekt skupine in deli tajnopis zasebnega ključa skupine z novim članom.
Sprememba skupine – odstranjen član	1. Izdelaj javni in zasebni ključ. 2. Odstrani člana iz objekta skupine. 3. Digitalno podpiši objekt skupine. 4. Shrani objekt skupine in deli tajnopis zasebnega ključa s člani nove skupine.

Tabela 3.1: Operacije skupine

3.3.3 Uporabniki

Storitev za upravljanje z uporabniki

Za zagotavljanje informacij o uporabnikih in njihovo upravljanje je zadolžena storitev, imenovana *UserService*. Poleg skupnih lastnosti vseh storitev je ta storitev v vlogi posredovalke javnih ključev uporabnikov ostalim storitvam (*GroupService* na primer potrebuje javni ključ novega dodanega člana).



Slika 3.8: Komponenta za prikaz uporabnikov

Podatkovni tip uporabnika sestavljajo naslednji atributi:

- ID uporabnika
- ime uporabnika,
- priimek uporabnika,
- ID-ji skupin, h katerim spada, in

- javni ključ uporabnika.

Pregled uporabnikov

Do komponent za pregled uporabnikov (slika 3.8) in dodajanje/urejanje uporabnika lahko pride skrbnik.

Izdelava uporabnika

Ko skrbnik vnese podatke uporabnika, komponenta preveri veljavnost podatkov (prazna polja in veljavni format e-naslova) ter sproži shranjevanje. Postopek shranjevanja nato prevzame storitev *UserService*. Ob določitvi novega uporabnika ta še nima zasebnega in javnega ključa, torej ga skrbnik še ne more dodati v skupine. Novi uporabnik je lahko dodan v skupine šele, ko prvič uporabi spletno aplikacijo (odpre povabilo v e-poštnem sporočilu). Ob prvi uporabi si uporabnik nastavi geslo, spletna aplikacija pa v ozadju izdela javni in zasebni ključ za uporabnika. Šele takrat ga lahko skrbniki dodajo v želene skupine kot člana. Primer medmrežne komunikacije pri določanju uporabnika je razviden s slike 3.7.

Urejanje uporabnika

Če spreminjamo samo podatke o uporabniku (ime, priimek ...), storitev spremembe preprosto pošlje zaledni storitvi. Če pa je bilo članstvo v skupinah, h katerim uporabnik spada, spremenjeno (dodano ali odvzeto), se sproži ustrezna akcija v storitvi *GroupService* (spremeni se objekt skupine, ne objekt uporabnika, saj skupina nosi informacijo, kdo je član).

Odstranjevanje uporabnika

Ko skrbnik sproži izbris uporabnika, je ta akcija izvršena na podlagi storitve *UserService*. Ta mora poleg fizičnega izbrisa uporabnika iz zbirke podatkov, uporabnika odstraniti še iz vseh skupin, katerih član je bil. Kljub temu da smo uporabnika izbrisali, še vedno pozna zasebni ključ skupine in zasebne ključe gesel, zato moramo vse te ključe zamenjati. V nasprotnem primeru je sicer res, da zaledna storitev uporabniku ne bi posredovala tajnopisa ključev,

do katerega nima več dostopa, a če bi uporabniku bil omogočen fizični dostop do zbirke podatkov, bi lahko neposredno dešifriral tajnopis gesel s ključem, ki ga je poznal od prej. Zato storitev *UserService*, še preden izbriše uporabnika, kliče operacijo po odstranitvi uporabnika iz vseh skupin. To operacijo izvaja storitev *GroupService* (postopek odstranjevanja uporabnika iz skupine).

3.3.4 Gesla

Storitev za upravljanje z gesli
Za zagotavljanje informacij o geslih in njihovo upravljanje je zadolžena storitev, imenovana *PassService*.

Geslo je v tem kontekstu mišljeno kot skupek prijavnih informacij in lahko vsebuje karkoli. Geslo za SSH-dostop do strežnika bi na primer vključevalo:

- IP-naslov: 10.1.80.16,
- uporabniško ime: JanezNovak,
- geslo: 123.

Sam podatkovni tip gesla, ki ga vidi in hrani zaledna storitev, je sestavljen iz naslednjih atributov:

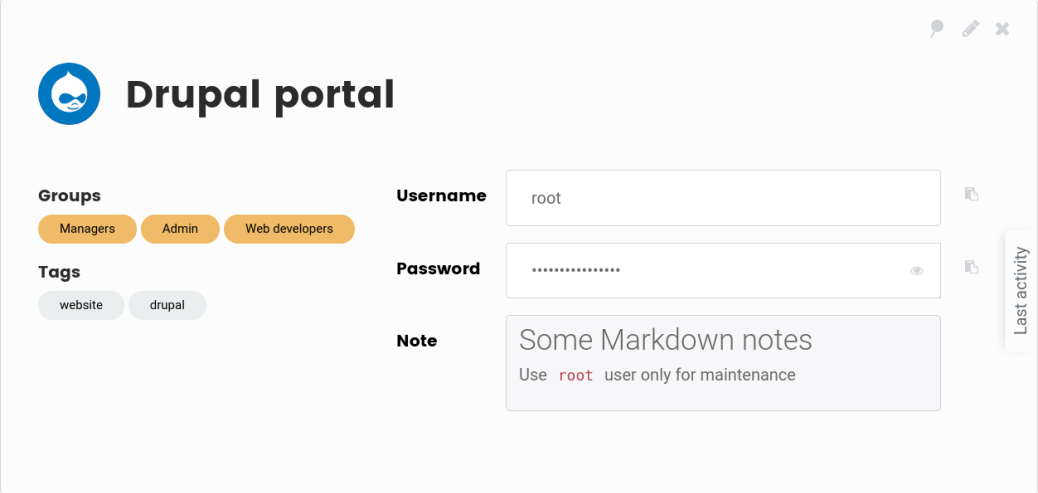
- ID-ja gesla,
- metapodatkov – imena, ikone, različice, seznama oznak in
- tajnopisa gesla.

Podatkovni tip gesla, ko ga dešifrira spletna aplikacija, je sestavljen iz naslednjih atributov:

- ID-ja gesla,
- metapodatkov – imena, ikone, različice, seznama oznak in
- seznama podatkov gesla (na primer IP-naslov, uporabniško ime, geslo).

Metapodatki so javni in niso šifrirani, namenjeni pa so zagotavljanju opisa, kaj geslo vsebuje. To je v pomoč predvsem pri iskanju gesel, saj

na ta način lahko že zaledna storitev filtrira gesla, ne da bi prebrala njihovo vsebino. Prav tako pa spletni aplikaciji ni treba dešifrirati vsebine takoj ob prikazu, temveč le, ko uporabnik to zahteva. Iz tega razloga je tudi izmed skupnih lastnosti vseh storitev vmesnik *CanActivate* implementiran nekoliko drugače. Iz zaledne storitve namreč ne prenese polnih objektov gesel, ampak samo objekt z metapodatki (brez tajnopisov). Šifropis gesla se prenese v aplikacijo, če je treba, ko uporabnik želi odpreti geslo (komponenta za prikaz gesla je prikazana na sliki 3.9).



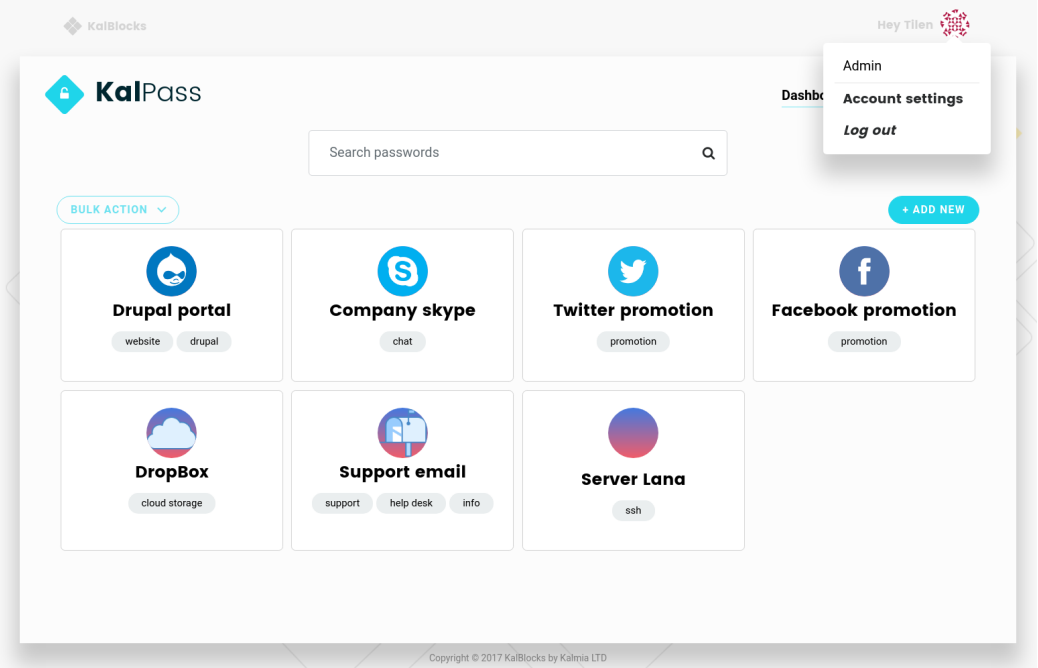
Slika 3.9: Komponenta za prikaz gesla

Pregled gesel

Pri pregledu gesel (slika 3.10) uporabnik vidi le gesla, ki jih lahko odpre (je član v ustrezni skupini). V tem stanju ima storitev *PassService* le seznam vseh gesel z metapodatki, brez tajnopisov. To je dovolj, da prikaže osnovne informacije in omogoča hitro iskanje. Ko uporabnik želi odpreti geslo, se iz zaledne storitve pretoči tajnopis za izbrano geslo, ki ga nato storitev dešifrira z zasebnim ključem skupine.

Dodajanje gesla

Že ob odprtju obrazca za dodajanje novega gesla so samodejno dodane vse



Slika 3.10: Komponenta za prikaz gesel

skupine, katerih uporabnik je član. Uporabnik lahko nato odstrani skupine, s katerimi ne želi deliti gesla. Pogoj je le, da je geslo deljeno z vsaj eno skupino, katere član je uporabnik, ki določi geslo. V nasprotnem primeru podobno, kot velja ob izdelovanja skupine, gesla ni mogoče določiti, saj bi to pomenilo, da uporabnik, ki nima pravice dostopa do gesla, pozna njegov dešifrirni ključ.

Ko uporabnik sproži dodajanje na novo vnesenega gesla, shranjevanje prevzame storitev *PassService*. Za geslo izdela nov zasebni ključ, s katerim šifrira vsebino. Izdelan ključ zašifrira za vsako skupino posebej z njenim javnim ključem. Nato celoten objekt gesla skupaj s tajnopisi zasebnega ključa za dodane skupine pošlje zaledni storitvi.

Urejanje gesla

Dodajanje skupine

Ko dodamo novo skupino, preprosto šifriramo zasebni ključ gesla z javnim ključem skupine.

Odvzemanje skupine

Ker se tu zgodi ponovno šifriranje podatkov in posodabljanje v vseh skupinah, moramo najprej preveriti, ali šifriranje z javnim ključem še da enak rezultat, kot je trenutno shranjen v skupini. Če ne, pomeni, da je nekdo spremenil javni ključ skupine (prišlo je do napada).

Reševanje sočasnega spreminjanja gesel

Vsak zahtevek za spremembo gesla, vsebuje tudi verzijo gesla na podlagi katerega je uporabnik spreminjal vsebino. Namen tega je preprečitev prepisovanja vsebine med uporabniki. Na primer uporabnik A in uporabnik B urejata neko geslo. Uporabnik A shrani vsebino na spletni strežnik (verzija gesla se poveča za 1). Za njim isto stori uporabnik B. Spremembe uporabnika B bodo zavrnjene, ker se verzija gesla ne bo ujemala z obstoječo.

Brisanje gesla

Pri brisanju gesla ni potreben noben dodatni korak. Zaledna storitev izbriše geslo in tajnopise zasebnega ključa gesla vsem skupinam, ki so ga vsebovale.

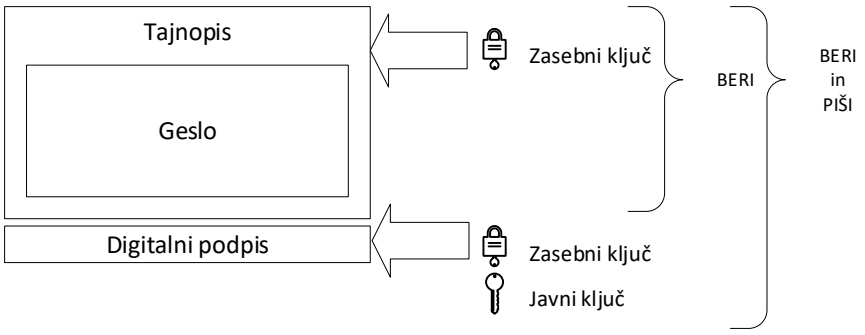
Tabela 3.2 povzema operacije za opisana dejanja nad gesli.

dejanje	**operacije**
Izdelava gesla	1. Izdelaj zasebni ključ. 2. Pridobi javne ključe skupin, h katerim bo spadalo geslo. 3. Shrani tajnopis gesla in deli tajnopis zasebnega ključa s skupinami.
Sprememba gesla – dodana skupina	1. Pridobi javni ključ dodane skupine. 2. Deli tajnopis zasebnega ključa gesla z dodano skupino.
Sprememba gesla – odstranjena skupina	1. Izdelaj nov zasebni ključ. 2. Pridobi javne ključe skupin, h katerim spada geslo. 3. Shrani tajnopis gesla in deli nov tajnopis zasebnega ključa s skupinami.

Tabela 3.2: Operacije gesel

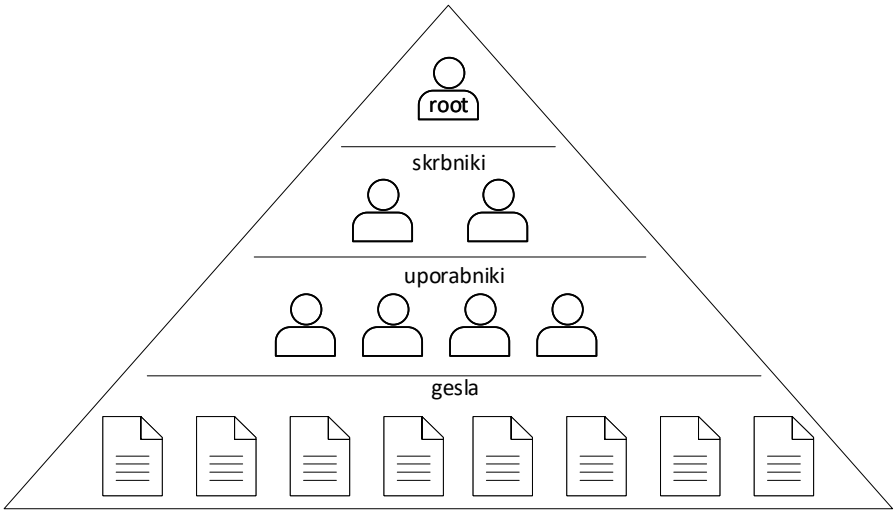
3.4 Pomanjkljivosti in izboljšave

Trenutno implementacija ne omogoča nadzora, ali bo imela neka skupina samo pravico brati ali tudi spreminjati gesla. Če je uporabniku omogočen dostop do gesla, ga ima samodejno tudi pravico spreminjati. To bi lahko rešili z dodatnim zasebnim in javnim ključem za vsako geslo. Skupine bi poleg zasebnega ključa za geslo hranile tudi dodatni zasebni in javni ključ namenjen digitalnemu podpisovanju gesla (slika 3.11). Če skupina pozna zasebni ključ za digitalno podpisovanje gesla, pomeni, da ga lahko podpiše in s tem dokaže pravico do spreminjanja gesla. Tako uporabniki skupin, ki ne poznajo zasebnega ključa za digitalni podpis gesla, ne morejo digitalno podpisati sprememb gesla.



Slika 3.11: Geslo z dodatnima ključema za uvedbo pravice spreminjanja

V aplikaciji uporabnike delimo na navadne uporabnike in skrbnike. Ker pa je zanašanje samo na skrbnika, da nikoli ne bo pozabil svojega gesla za dostop, zelo tvegano, bi lahko vpeljali dodatnega uporabnika *root*, ki bi bil samodejno dodan vsem skupinam in ga ne bi bilo mogoče odstraniti (slika 3.12). Takega uporabnika bi lahko podjetje uporabilo v skrajni sili, ko ne bi moglo več dostopati do gesel s svojimi računi (ker je bilo prijavno geslo pozabljeno ali pa je zaposleni nedosegljiv ...). Uporabnik *root* bi bil določen že ob začetku



Slika 3.12: Hierarhija uporabnikov

postavitve aplikacije. Njegov zasebni ključ bi bil natisnjen na papir formata A4 in shranjen v sefu podjetja. Lahko bi rekli, da je uporabnik *root* kot hladna shramba (angl. *cold storage*), saj ni mišljen za običajno uporabo. Prav tako bi ta račun lahko bil uporabljen ob nesrečah, ko je treba pridobiti neko geslo iz varnostnih kopij.

Poglavje 4

Sklepne ugotovitve

V diplomskem delu sta predstavljena postopek implementacije šifriranja podatkov znotraj spletne aplikacije (SPA), ter način varne izmenjave podatkov med uporabniki. Za predlagano rešitev je bila razvita spletna aplikacija KalPass. Ob razvoju aplikacije so pri opisovanju poudarjeni postopki, na katere moramo biti posebej pozorni, saj je ob površni implementaciji celotna zaščita podatkov ogrožena.

Danes se soočamo s številnimi spletnimi napadi na zaledne storitve. Zato menimo, da je nujno pri načrtovanju aplikacije in njenega zaledja vključiti čim več varnostnih plasti, kajti ne glede na stopnjo varnosti vedno obstaja možnost zlorabe. Tudi če ima na primer podjetje svojo infrastrukturo postavljeno v DMZ (angl. *demilitarized zone*), ima lahko slabo fizično varovanje infrastrukture ali pa zaposleni podležejo napadu socialnega inženiringa.

Opisani pristop se lahko uporabi v vseh spletnih aplikacijah, kjer se zahteva zaščita podatkov. Teh pa ne potrebuje nihče drug kot uporabniki aplikacije. Če zaledna storitev potrebuje podatke v dešifrirani obliki (vodenje statistike, nadaljnje akcije ...), opisana zaščita ni ustrezna, saj tega ne omogoča. Prav za slednje bi v nadaljevanju lahko raziskali še možnosti analize podatkov, brez razkritja njihovih informacij (dešifriranja).

Predstavljena rešitev učinkovito ščiti podatke pred takojšnjim nepooblaščenim dešifriranjem podatkov. Žal se je izkazalo, da moramo storitvi za zagotavljanje javnih ključev zaupati. V nasprotnem primeru je integriteto javnih ključev nemogoče jamčiti. Če napadalcu uspe zamenjati javni ključ s svojim, lahko članstvo v skupini ponaredi. S tem napadalcu legitimni uporabniki lahko nevede dodelijo tajni ključ do podatka. Težava pri tovrstnem napadu je, da mora priti do ponovnega šifriranja podatka, kar pomeni, da do uspešne pridobitve ključa lahko preteče kar nekaj časa, s časom pa se možnost, da bo spremembo nekdo opazil, poveča.

Literatura

[1] J. Pelzl C. Paar. *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin, Heidelberg: Springer, 2009.

[2] M. Naehrig E. Wustrow J. W. Bos J. A Halderman N. Heninger, J. Moore. *Elliptic Curve Cryptography in Practice*. Dostopno na: `https://eprint.iacr.org/2013/734.pdf`, 2013. (pridobljeno 22. 7. 2017).

[3] K. Maletsky. *RSA vs ECC Comparison for Embedded Systems*. Dostopno na: `http://www.atmel.com/Images/Atmel-8951-CryptoAuth-RSA-ECC-Comparison-Embedded-Systems-WhitePaper.pdf`, 2015. (pridobljeno 23. 7. 2017).

[4] A. Harrington C. D. Jensen. *Cryptographic Access Control in a Distributed File System*. Dostopno na: `https://pdfs.semanticscholar.org/8f7c/5054aafbe2892e4323a679631d8e31f30eb4.pdf`. (pridobljeno 1. 8. 2017).

[5] Google Inc. Angular, uradna stran. Dostopno na: `https://angular.io`. (pridobljeno 19. 5. 2017).

[6] B. Suresheb. *Implement an SPA with Angular 2*. Dostopno na: `https://www.ibm.com/developerworks/library/wa-implement-a-single-page-application-with-angular2/index.html#N10091/`, 2016. (pridobljeno 15. 7. 2017).

[7] C. Nance. *TypeScript Essentials*. Community experience distilled. Packt Publishing, 2014.